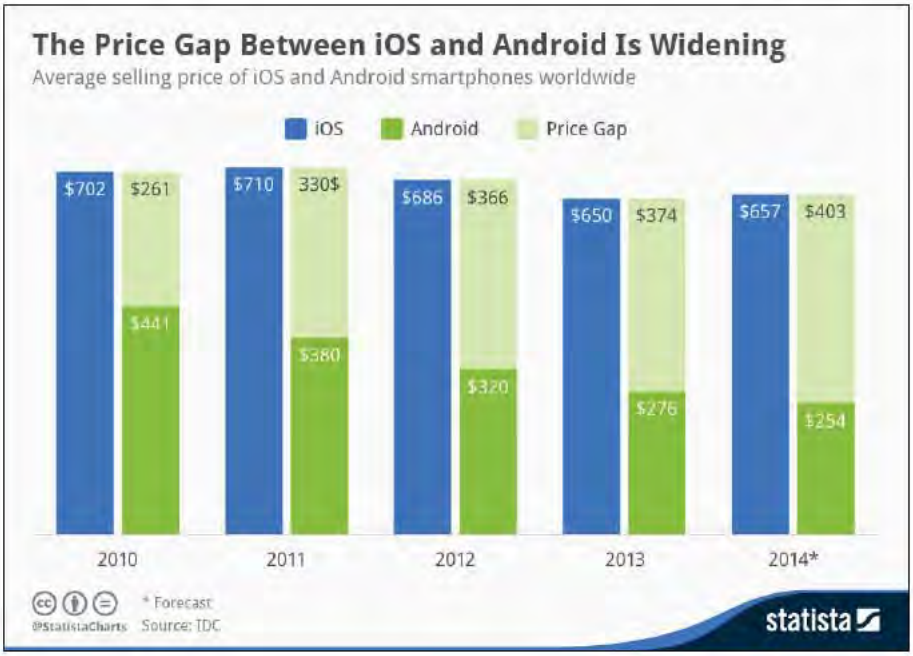
EXHIBIT G
PART 2 OF 6

REDACTED VERSION OF DOCUMENT SOUGHT TO BE SEALED

Figure 5

Average Annual iPhone and Android Smartphone Prices – 2010 to 2014[82]



89. Nowhere in the Leonard Report does Dr. Leonard discuss or evaluate the $261 to $403 price difference between iPhones and Android smartphones, or how this significant difference impacts his conclusion that 41 to 44 percent of Android Ad Revenue would have been "diverted" to users of iPhones during the eight-year period 2008 to 2015.[83] Without such an analysis, Dr. Leonard's conclusions lack merit.

4.5.5.2 Dr. Leonard Fails to Provide Substantiating Analysis for Diversion Percentages

90. Leonard Exhibit 1b – iPhone Recapture Adjustment – utilizes annual "Diversion Ratios" of 40.5 to 44.0 percent to derive an "iPhone Recapture Adjustment" of [redacted] According to Dr. Leonard, "[b]ased on an analysis discussed below, I determined that Google would recapture at least 44% of its ad revenue on Android handsets with ad revenue on iPhones. Applying this recapture rate to Android ad revenue yields the incremental ad revenue that Google would have made on the iPhone in the absence of Android."[84] Notes to Leonard Exhibit 1b refer to Leonard 3d.2 "[f]or the diversion ratio."

---

[82] The Price Gap Between iOS and Android is Widening, Statista, Felix Richter, June 1, 2014; https://www.statista.com/ chart/1903/average-selling-price-of-android-and-ios-smartphones/

[83] Expert Report of Dr. Leonard, February 8, 2016, Exhibit 1b.

[84] Expert Report of Dr. Leonard, February 8, 2016, p. 27.

91. Nowhere else in the Leonard Report does Dr. Leonard analyze, evaluate, or discuss how the "diversion ratios" of 41 percent to 44 percent are derived. Without such an analysis and discussion, Dr. Leonard's opinions and conclusions are unsubstantiated and therefore unreliable.[85]

4.5.6 Dr. Leonard Substantially Understates "Android Profits" (see Leonard Ex. 1a.4)

92. Dr. Leonard's "top-down" "Android Profits" approach begins in Leonard Exhibit 1a.4 – Profit Apportioned to Android Versus Search/Ads Technologies and Services – and continues to Leonard 3e – Top Down Apportionment. This approach results in "Android Profit" of [redacted] and "Android Profit (Apportioned to the 37 APIs)" of $32.4 million.[86] Dr. Leonard's "top-down" "Android Profits" approach is defective for several reasons including at least the following.

4.5.6.1 Dr. Leonard's [redacted] TAC Savings Factor is Too Low

93. As illustrated in Leonard Exhibit 1a.4, Dr. Leonard utilized a cost-based metric of [redacted] to quantify the portion of Android Search Ad Revenue that Dr. Leonard asserts is attributable to the Android platform. The [redacted] figure is derived from financial data reflected in a May 2015 Google presentation entitled "Introduction to Android."[87] Based on that financial information, Dr. Leonard concludes that Google earns [redacted] less profit for Search Ad Revenue generated from iPhones than it does from Search Ad Revenue generated from Android devices.[88]

94. Dr. Leonard's [redacted] figure is based on the difference between an imputed Search Ad Revenue-per-Android-unit figure of [redacted], and an imputed net margin from Search Ad Revenue-per-iPhone figure of [redacted] The May 2015 Google presentation attributes the difference to the TAC Google pays to Apple Inc. to direct Internet traffic from iPhones and iPads to Google websites.[89] Dr. Leonard's [redacted] TAC savings factor is too low.

95. On January 20, 2016, this Court entered an Order Re: Motion to Compel,[90] whereby Google was ordered to produce "charts specifying the following data for each provider of a non-Android mobile operating system with whom Google has or previously had a search distribution agreement for Google search services offered in connection with such non-Android mobile operating system, where such agreement also provided for the sharing of revenue."[91]

96. In response to that Order, Google produced a document entitled "Google Search Distribution Agreements with Non-Android Mobile Operating System Partners" ("Google's Non-Android

---

[85] Dr. Leonard's iPhone Recapture Adjustment set forth in Leonard Exhibit 1b also suffers from some of the same defects I addressed above. Namely, Android-related TAC of [redacted] is too high, and the "Incremental Search and Advertising Expenses" are improperly allocated to the Android platform.

[86] Expert Report of Dr. Leonard, February 8, 2016, Exhibit 3e.

[87] GOOG-00130338-386 at 343.

[88] Expert Report of Dr. Leonard, February 8, 2016, p. 33; [redacted]

[89] GOOG-00130338-386 at 343.

[90] Order Re: Motion to Compel, 3:10-cv-03561-WHA, Docket No. 1436, January 20, 2016, p. 2.

[91] Order Re: Motion to Compel, 3:10-cv-03561-WHA, Docket No. 1436, January 20, 2016, p. 1.

Mobile O.S. Partner List"). For six Google non-Android Mobile O.S. Partners, this document provides: 1) the percentage of Search Revenue Google shares with the partner; 2) the total gross revenue earned by Google under the agreement; and 3) the Google search services which are the subject of the respective agreements.

97. Exhibit 7.6 is a calculation of the weighted average percent of Search Revenue Google shares with all six Partners reflected in Google's Non-Android Mobile O.S. Partner List. As Exhibit 7.6 illustrates, since 2006, Google has paid back to its non-Android Mobile Operating System Partners [redacted] of the Search/Ad Revenue it earns from non-Android mobile devices of those partners.

98. Given the availability of the information in Google's Non-Android Mobile O.S. Partner List, it is curious that Dr. Leonard chose to rely only on imputed profit margin figures reflected in one Google presentation that compares the annual profitability of Android devices to only iPhones as of a certain point in time (2015). In any event, the [redacted] factor derived by Dr. Leonard is about 3.6 percentage points lower than the average TAC percentages of Search Ad Revenue Google paid to non-Android Mobile O.S. Partners during the relevant time period.

99. Dr. Leonard's application of his imputed [redacted] TAC savings to the [redacted] of Search Revenue Google earned from Android devices during the years 2008 to 2015 results in the understatement of the Android Search Ad Revenue attributable to the Android platform.

4.5.6.2 Google Would Not Have Realized 100% of AdSense/Display Revenue Absent Android

100. AdSense and Display revenues are generated from Android devices. However, Leonard Exhibit 1a.4 excludes revenue, TAC, and profit for the AdSense and Display Ad Revenue Google realized from Android devices. Presumably, it is Dr. Leonard's position that no AdSense or Display revenue, TAC or profit is attributable to the Android platform, or conversely, that all AdSense and Display Ad Revenue, TAC and profit is attributable to Search Ads Technologies and Services. If this is, in fact, Dr. Leonard's position, a necessary underlying assumption is that Google would have achieved 100 percent of the AdSense and Display Ad Revenues it realized through Android devices, regardless of whether or not the Android operating system was developed and commercialized. If this is Dr. Leonard's position, it is undermined by several factors.

101. First, Google considered the risks that it faced of being locked out of other platforms, and considered those risks to be so material that it would undertake a long-term strategy of building its own platform. Google necessarily concluded either that the risk to achieving the revenue, or the risk of a dramatically increased cost of acquiring the revenue, was so high that a lengthy and expensive build-your-own approach was warranted. Even after it was faced with this lawsuit, Google's President of Platforms and Mobile Media concluded that the risk to Google of missing

the "critical mobile window" was an existential one—stating that Google will be "out of business in 10 years" if it does not capture the mobile business.[92]

102. This lockout/control issue applies to display advertising as well as other types. For example, modern web browsers have built-in capabilities, or add-on extensions, that allow for ad-blocking. The Safari "Reader View" is an ad-blocking capability.[93] Microsoft also has a built-in ad-suppression feature for its Edge web browser called "Reading View."[94]

103. In addition, I understand that AdSense and Display Ad Revenues are influenced by Android user data collected by Google. I understand that Google utilizes user data collected from Android devices to direct relevant AdSense and Display advertising to Android users and/or to direct users to Google and Google Network Partner websites. This results in increased Display and AdSense Ad Revenues. Absent this Android user data, Google would not achieve the same levels of AdSense and Display revenues and profits.

104. Third, without its own dominant position as a mobile platform provider, Google could not be assured of the same favorable terms for traffic acquisition on competitive platforms.

105. In **Sections 4.2-4.4** above, I set forth the limitations of Dr. Leonard's "iPhone Recapture Adjustment." Those limitations undermine Dr. Leonard's presumed assertion that some portion of the AdSense and Display Ad Revenue realized from Android devices would have been realized through iPhones in the absence of Android devices.

106. In addition to the limitations of Dr. Leonard's "Diversion Ratio" theory, his presumed theory that the remaining (56 percent) portion of Android-generated AdSense and Display Ad Revenue would have been "diverted" to non-iPhone devices in the absence of Android lacks merit.

4.5.7 Dr. Leonard's General & Administrative Expense Allocation is Improper

107. Like his "top-down" "Android-Related Profits" analysis contained in Leonard Exhibit 1a.1, Dr. Leonard's "top-down" "Android Profits" analysis reflected in Leonard Exhibit 1a.4 improperly allocates "Android General and Administrative" expenses of [redacted] to the Android platform.[95] This allocation is improper for the same reasons set forth in Section 4.5 above.

4.6 Dr. Leonard's Top-Down ("Lines of Code") Apportionment Methodology is Unreliable

108. The "top-down" apportionment methodology reflected in the Leonard Report is purportedly based on the percentage of Android's total lines of source code represented by the Infringed Java

[92] GOOGLE-23-00000049.
[93] http://fortune.com/2015/09/22/ad-block-ios-android/.
[94] http://www.engadget.com/2015/07/30/microsoft-edge-windows-10/
[95] Expert Report of Dr. Leonard, February 8, 2016, Exhibit 1a.1.

Copyrights. Such an apportionment methodology is invalid, and has been rejected by federal courts.[96] A straight "lines of code" allocation does not account for the *value* that the Java APIs offered to the platform. For example, according to the United States District Court for the Northern District of California in an unrelated matter, "[a]lthough A10 points to the ratio between the 145 lines of infringing code and the 10 million lines of code in Brocade's product, that ratio fails to account for the evidence suggesting the importance of the implementing code to Brocade's software."[97]

109. Apportionment is supposed to assign merit to the relative contribution between the infringing and noninfringing portions in generating the profits at issue. In other words, the apportionment methodology is supposed to be a reasonable approximation of the value contributed by the copyrighted material. Value takes into account many factors other than just lines of code.

110. Google's apportionment approach is invalid because the percentage of total lines of Android source code represented by the Infringed Java Copyrights is not indicative of the value of the Infringed Java Copyrights.

111. Moreover, according to the lines of code technical analysis performed by Oracle's expert Dr. Schmidt, a significant portion of the lines of code in Android include contributions from other third parties, blank lines, comments, and unspecified code authorship on which Google has not affixed a copyright notice.[98] Dr. Leonard's analysis implicitly assigns equal value to every single one of these lines of code. Evidence that demonstrates that the value of the Infringed Java Copyrights far exceeds that suggested by Dr. Leonard's apportionment methodology is discussed in the following subsections.

4.6.1 Android Does Not Work Without the Infringed Java Copyrights

112. I understand that the Android platform is critically dependent on the 37 Java APIs, both individually and collectively. According to Dr. Schmidt, the build process fails if the files from even one of the infringing 37 Java APIs are removed. As a result, Android will not run on a mobile device without all of the copied declaring code and the full set of files for the 37 Java APIs.[99] According to Dr. Schmidt, Android is not usable on a computing device, such as a phone or tablet, without each of the 37 Java APIs, or the copied declaring code in them.[100]

---

[96] See, Computer Associates Int'l, Inc. v. Altai, Inc., 775 F.Supp. 544, 571 - 572 (E.D.N.Y. 1991); Brocade Communications Systems, Inc. v. A10 Networks, Inc., 2013 WL 831528 (N.D.Cal).

[97] Brocade Communications Systems, Inc. v. A10 Networks, Inc., 2013 WL 831528 (N.D.Cal).

[98] Expert Report of Dr. Schmidt, February 29, 2016, Figures 1 and 2.

[99] Expert Report of Dr. Schmidt, January 8, 2016, ¶ 121.

[100] Expert Report of Dr. Schmidt, January 8, 2016, ¶ 78

113. Conversely, I understand that many of the other Android APIs could be removed from the Android source code without causing the build process to fail.[101] This is because most of the other APIs are not part of the Android core library.[102] The fact that the Android build process fails if any one of the 37 Java APIs is removed from the Android source code indicates that the Infringed Java Copyrights are relatively more valuable than other Android platform components including other Android APIs and other segments of the Android source code.

4.6.2 The Infringed Java Copyrights Provided Stability to the Android Core Library

114. As indicated in my Initial Report, Mr. Reto Meier, an Android developer advocate at Google since 2009, testified that Google copied the core Java APIs into Android instead of creating its own because "utilizing the same [Java APIs] would make it easier for folks to -- to use [Android] if they had experience with [the Java APIs]."[103] Bob Lee, head of Android's core library team at Google, agreed in his deposition that the 37 APIs "are [the] good stuff from Java."[104] Dan Bornstein, technical lead of Android's Dalvik virtual machine and core libraries team at Google, agreed in his direct examination that, "absolutely," his "determination of what packages would be implemented in the core library" was related to his "expectations of Java language programmers."[105]

115. According to Dr. Kemerer, the 37 Java APIs represent 73 percent of the stable Android core library.[106] According to Dr. Kemerer, the 37 Java APIs render the Android platform 82 percent more stable.[107] I also understand that 37 Java APIs stabilized around 10.9 years after the first release of the JDK whereas the Android Core APIs stabilized after only 2.3 years after the first release of Android.[108] Had the Android Core libraries taken 10.9 years to stabilize, Google would have missed the timing window to enter the mobile handset market when it did and would have been unable to launch the Android handset on time. Thus, the relative importance of the Android core library to the Android platform, and the significance of the Infringed Java Copyrights to the stability of the Android platform indicate that the Infringed Java Copyrights are more valuable than a value indicated by an approach based on a percentage-of-lines-of-source-code ratio, such as that utilized by Dr. Leonard.

4.6.3 The 37 Java APIs are Called Upon More Often by Popular Mobile Applications

---

[101] Expert Report of Dr. Schmidt, February 29, 2016, ¶ 76.
[102] See Section 6 below.
[103] Deposition of Reto Meier, December 11, 2015, p. 113.
[104] Deposition of Bob Lee, August 3, 2011, p48.
[105] Trial Testimony of Daniel Bornstein, Transcript Vol. 08, April 25, 2012, p. 1782.
[106] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 108.
[107] Expert Report of Dr. Kemerer, February 29, 2016, ¶ 27.
[108] Expert Report of Dr. Kemerer, February 29, 2016 ¶ 17.

116. According to Dr. Kemerer, an analysis was conducted to assess the relative importance of the 37 Java APIs in the Android App ecosystem from the perspective of App developers.[109] According to Dr. Kemerer, "the initial analysis empirically demonstrates the extent to which developers depend on the declarations provided by the infringed packages in the development of Android apps."[110]

117. The analysis, as described in the Kemerer Report consisted of selecting the top 100 Apps for analysis.[111]

118. Based on this analysis, Dr. Kemerer concluded that:

> *[T]he 37 Java APIs are a critical requirement for essentially every one of the 100 top applications. In fact, every one (100%) of the top 100 apps depends upon a minimum of three of the 37 copied Java API packages. The average number of dependencies is 11.5, or nearly a third of the 37 copied API packages. And one of the top 100 apps depends on 23 of the 37 copied API packages.*[112]
>
> *If the analysis is restricted to the most popular of the 100 apps (the 14 apps that are listed as having between 1,000,000,000 and 5,000,000,000 downloads), they can be seen as being even more dependent upon the 37 copied API packages, with the minimum number of dependencies being eight, the average number 13.8, and the maximum number 17.*[113]

119. This analysis indicates that application level dependencies on the 37 Java APIs are significant, and central to the Android app ecosystem and its developers.[114]

120. Dr. Leonard asserted that "most games, for example are written in C++ because of the performance benefits inherent in avoiding the use of the virtual machine."[115] I understand from Dr. Kemerer's analysis of NDK Application Dependency, he found that contrary to Dr. Leonard's assertions, very popular apps and games are heavily dependent on significant numbers of the 37 Java APIs. Some of the applications analyzed include Candy Crush, Instagram, Snapchat, Twitter and Angry Birds. [116]

121. Thus, it appears that Dr. Leonard's assertions minimizing the importance of the 37 APIs to popular mobile applications is factually incorrect. Since the dependency on the 37 API is demonstrably significant, it is evident that the relative importance of the Infringed Java Copyrights

---

[109] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 126.
[110] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 126.
[111] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 134.
[112] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 135.
[113] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 136.
[114] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 140.
[115] Expert Report of Dr. Leonard, February 8, 2016, ¶ 110.
[116] Expert Report of Dr. Kemerer, February 29, 2016, Table 2: Dependence of Leonard-identified "NDK" Android apps on the 37 copied packages

to the Android app ecosystem and its developers indicates that the Infringed Java Copyrights are more valuable than the value indicated by an apportionment approach based on a percentage-of-lines-of-source-code ratio, such as the approach adopted by Dr. Leonard.

4.6.4 Google's Own Page Rank Score Demonstrates the Centrality of the 37 Java APIs

122. According to Dr. Kemerer, "centrality" is a metric that is used to describe the relative importance of a particular entity, or node, within a network of interconnected entities.[117] According to Dr. Kemerer, "[a]pplying the notion of network centrality to the 37 Java API packages within the context of the Android source code as a whole is one way to understand how important those packages are to Android."[118] According to Dr. Kemerer, a high centrality score for the 37 Java APIs would indicate that the classes inside of them are connected to a high number of classes in packages outside of those packages. Such a pattern would indicate that the rest of the Android source code heavily depends on the 37 Java APIs.[119]

123. I understand that "PageRank" is a centrality measure that was developed by Larry Page and Sergey Brin as part of their research to develop a new search engine.[120] According to Dr. Kemerer, PageRank is now a widely referenced tool for network analysis, and is an appropriate metric for evaluating the centrality of the 37 Java APIs in the Android source code.[121]

124. A PageRank analysis was undertaken to identify the extent to which the Android source code leverages the functionality provided by the 37 Java APIs.[122] The Android PageRank analysis produced a "PageRank score" for those classes which belong to the 37 Java APIs, as well as PageRank scores for the rest of the class groupings across the wider Android source code (i.e., the non-copied classes).[123]

125. Figure 11 of the Kemerer Report compares the average PageRank score of the classes which belong to the 37 Java APIs to the average PageRank score of other classes within the Android source code ("non-copied classes"). According to Dr. Kemerer, Figure 11 to his report "very clearly demonstrates the vastly greater PageRank scores of copied classes, with the average copied class boasting a score that is over 30 times greater than that of the average non-copied class."[124]

---

[117] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 141.
[118] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 143.
[119] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 144.
[120] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 146.
[121] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 148.
[122] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 150. I understand that the Android software system network used in this analysis considers every single Java class from Version 5.1.0, release 1 (Lollipop) of the Android Open Source Project (AOSP) source code.
[123] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 152.
[124] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 153.

According to Dr. Kemerer, "[t]hese PageRank results show the importance of the 37 copied Java API packages to the network of the Android operating system."[125]

126. The relative importance of the Infringed Java Copyrights to the network of the Android operating system indicates that the Infringed Java Copyrights are more valuable than the value indicated by an apportionment approach based on a percentage-of-lines-of-source-code ratio, such as the approach adopted by Dr. Leonard.

4.6.5 The 37 Java APIs Enabled Google to Get Android to Market More Quickly

127. According to Dr. Kemerer, when Android was first created, Google benefited by leveraging the popularity and familiarity of the Java platform (including the Infringed Java Copyrights) among developers in order to quickly attract them to the Android platform.[126] I understand that developers are generally most familiar with declaring codes and SSO elements, and less familiar with the underlying implementing code.[127] Google's use of the 37 Java APIs thus hastened the attraction of millions of developers' familiarity with the Android platform and assisted in Google's success in reaching and building a significant core of application developers.

128. Developers' familiarity with the Java platform, including the 37 APIs, not only attracted developers to the Android platform, but also enhanced the developers' productivity.[128] APIs allow for the faster, more efficient construction of high quality applications. Rather than engaging in the more laborious task of writing sequences of program code from scratch, developers were able to draw on the resources of the Java APIs, and use the packaged classes and methods to more easily create high quality programs.[129]

129. By using the familiar Java APIs, Google both attracted more developers to the Android platform and made the work of those developers easier, thus further accelerating the development and acceptance of the Android platform.[130] Google needed to develop a mobile platform quickly to establish its presence in the market and to start the process of monetizing data from user engagement with applications and devices.[131]

130. Google was motivated to get to this market quickly before competing mobile platforms gained significant market share at Google's expense, and before it lost the opportunity to dominate the

---

[125] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 153.
[126] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 64.
[127] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 76.
[128] Expert Report of Dr. Schmidt, January 8, 2016, ¶ 75.
[129] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 22.
[130] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 94.
[131] Expert Report of Dr. Kemerer, January 8, 2016, ¶ 66.

mobile platform so that it could generate revenue from advertising.[132] The record evidence illustrating Google's motivation is substantial, and includes the following:

- ***Sworn Testimony of Andrew Rubin:***

Q. And why did you want to accelerate your effort?

A. Why did I want to get to market faster?

Q. Yes.

A. Because I think it's a competitive advantage to come to market as quickly as possible.[133]

:::

Q. And what was the value of getting to market quickly to Google?

A. It gave us a competitive advantage.

Q. Against whom?

A. The existing incumbent platform manufacturers.

Q. Which were whom?

A. At the time, Microsoft, Symbian, RIM. Motorola had a couple of platforms.[134]

:::

Q. The deadline you were talking about, the December 2006 deadline, you said, "I was under incredible schedule pressure."

A. Yep.

Q. What did you mean by that?

A. Well, look, I mean, you have a window of opportunity in smartphones. I had competitors all over the place. When I started the company, Microsoft was my competitor. You know, there was Symbian in there as well, and, you know, all sorts of Linux initiatives. You have to ship as soon as feasibly possible.

I mean, you go to extraordinary length to ship sooner, because it's a very dynamic market. And it could shift directions at any time. Right. So my job as, you know, the architect of this business concept was to just do everything that I possibly could to get my solution to the market in the shortest time possible.[135]

---

[132] Dr. Schmidt Trial Testimony, April 24, 2012, 1456:15-19, 1457:19-25, 1458:1-16

[133] Deposition of Andrew Rubin, April 5, 2011, p. 38.

[134] Deposition of Andrew Rubin, April 5, 2011, p. 103.

[135] Deposition of Andrew Rubin, July 27, 2011, pp. 179-180.

- ***Sworn Testimony of Eric Schmidt:***

  Q. And one of the reasons that you were interested in having Android proceed as fast as it could was you wanted to beat Microsoft and Symbian to volume, correct?

  A. Yes.

  Q. And by beating Microsoft and Symbian to volume, you mean getting your handset out there with a lot of users before they had their handsets out there with a lot of user[sic]; is that fair?

  A. Yes. Volume means more users, so serving more customers.[136]

- ***A July 24, 2006 Andrew Rubin Email:*** indicates that Google was "in discussions for 8 months with Sun, walked away, and must prove that our internal effort is clean. Also, because we were in discussions for so long, we must acquire an existing implementation. We ship in 6 months!"[137]

- ***A January 2, 2006 Google Email*** : from Brian Swetland indicates that the "[r]easons to shift to a primarily Java API . . . simplifies the application development story . . . reduces our development time . . . faster app development and debuggability."[138]

- ***A 2006 Google Presentation***: describes the importance of leveraging Java developers and avoid the need to create a large developer services organization. According to the presentation, "[s]upporting Java is the best way to harness developers. Fact: Linux fragmentation threatens value. Tools and new app frameworks are biggest hurdles. 6M Java developers worldwide. Tools and documentation exist to support app development without the need to create a large developer services organization. There exist many legacy Java applications. The wireless industry has adopted Java, and the carriers require its support. Strategy: Leverage Java for its existing base of developers."[139]

4.7 Dr. Leonard's Bottom-Up (Cost Based) Apportionment Methodology is Unreliable

131. According to Dr. Leonard, "I understand that a plaintiff can seek 'unjust enrichment' damages, which are any profits of the infringer that are attributable to the infringement and are not taken into account in computing the plaintiff's actual damages."[140] Dr. Leonard cites to 17 U.S.C. §504(b) in support of this statement.[141] However, contrary to the opinion put forth by Dr. Leonard, "unjust enrichment" is not referenced in 17 U.S.C. §504(b) as a measure of monetary recovery for copyright infringement.

---

[136] Trial Testimony of Dr. Schmidt, April 24, 2012, p. 1458.
[137] Trial Exhibit 147, GOOGLE-01-00023889-890.
[138] Trial Exhibit 13, GOOGLE-01-00019511-513 at 513.
[139] GOOGLE-01-00025575-587 at 584.
[140] Expert Report of Dr. Leonard, February 8, 2016, pp. 7-8.
[141] 17 U.S.C. §504(b).

132. 17 U.S.C. §504(b) specifically states as follows:

> *The copyright owner is entitled to recover the actual damages suffered by him or her as a result of the infringement, and any profits of the infringer that are attributable to the infringement and are not taken into account in computing the actual damages. In establishing the infringer's profits, the copyright owner is required to present proof only of the infringer's gross revenue, and the infringer is required to prove his or her deductible expenses and the elements of profit attributable to factors other than the copyrighted work.*[142]

133. As 17 U.S.C. §504(b) indicates, the measures of monetary recovery for copyright infringement include the actual damages suffered by the copyright owner "and any profits of the infringer that are attributable to the infringement and are not taken into account in computing the actual damages."[143] Nowhere does the statute mention the concept of "unjust enrichment".

134. The American Institute of Certified Public Accountants ("AICPA") represents the CPA profession nationally regarding rule-making and standard setting. The AICPA develops professional standards and monitors and enforces compliance with the professional's technical and ethical standards.[144] Technical Practice Aids are a source of various authoritative and non-authoritative or implementation guidance issued by the AICPA and other organizations. Technical Practice Aids include questions and answers issued by the AICPA on a variety of accounting, auditing and industry topics.[145]

135. In 2013, the AICPA issued a Practice Aid entitled "Calculating Intellectual Property Infringement Damages" which sets forth the standards for calculating monetary recovery for copyright infringement. According to the AICPA:

> *17 USC 504 authorizes courts to grant copyright owners actual damages suffered as a result of the infringement. In addition, any profits of the infringer attributable to the infringement are granted to the copyright owner in order to remedy the damages caused by the infringement, as long as these damages are not duplicative. Should the copyright owner be unable to prove actual damages or the defendant's profits, 17 USC 504 alternatively grants the copyright owner the right to elect . . . to recover an award of statutory damages instead of actual damages and profits.*[146]

136. As seen above, the AICPA Practice Aid does not mention "unjust enrichment" in connection with copyright infringement. And furthermore, the only mention of "unjust enrichment" as a form of monetary recovery in the AICPA Practice Aid is with respect to monetary recovery for trade secret

---

[142] 17 U.S.C. §504(b).

[143] 17 U.S.C. §504(b).

[144] http://www.aicpa.org/about/missionandhistory/pages/missionhistory.aspx

[145] http://www.aicpa.org/publications/accountingauditing/techpractaids/pages/technicalpracticeaids.aspx

[146] Forensic & Valuation Services Practice Aid – Calculating Intellectual Property Infringement Damages, AICPA, 2013, p. 20.

misappropriation.[147] The Uniform Trade Secrets Act specifically references "unjust enrichment" in its remedies provisions.[148] Thus, Dr. Leonard's approach to calculating the measure of monetary recovery in this case is not only inconsistent with 17 U.S.C. §504(b), but it is also inconsistent with highly recognized standard setting organizations which provide guidance on its implementation.

137. With specific regard to Dr. Leonard's opinions, the Leonard Report includes three "bottom-up" approaches which attempt to measure the "cost savings" enjoyed by Google as a result of its illegal use of the Infringed Java Copyrights. Avoided costs can be a measure of unjust enrichment (as can a head start benefit calculation), but it is not a measure of actual profit disgorgement. Avoided costs is a way of measuring a different benefit received by the defendant—the benefit of avoiding a cost that should have been paid for use of the intellectual property at issue. In this case, for example, such a benefit might be characterized by whatever license fee Google did not have to pay to Sun and Oracle for using the Infringed Java Copyrights. Here there was no such license and so the avoidance of that payment would be entirely hypothetical. It is my understanding that in copyright cases such hypothetical, or constructive licenses, are considered a form of actual damages to the plaintiff rather than profit disgorgement by the defendant. From a financial perspective, the hypothetical license is simultaneously lost revenue to the plaintiff and a cost foregone by the defendant. This has nothing to do with profits earned by the defendant because of infringement. My opening report sets forth no hypothetical license fee as part of Oracle's claim for damages in this case.

138. In connection with preparing his three "bottom-up" approaches, Dr. Leonard analyzes a "counterfactual" (i.e. "but-for") world and concludes that Google would have incurred certain costs had it not used the Infringed Java Copyright in connection with its development of Android. Although Dr. Leonard opines that "[t]he appropriate measure of the apportionment of Google's Android-related profits to the alleged infringement using the bottom-up approach is the minimum among the three cost-savings and the profit loss,"[149] his cost savings metrics clearly do not reflect an apportionment of Google's causally connected profits. Rather, they improperly reflect an unjust enrichment theory of damages. In the paragraphs that follow, I provide a further critique of the analytics and related assumptions underlying Dr. Leonard's "bottom-up" approaches to apportionment.

139. Dr. Leonard's use of cost savings to apportion Google's causally connected profits related to infringing attributes of the Android Platform is unsupported by case law. Specifically, in his "bottom-up" apportionment, Dr. Leonard calculates the "generated cost savings for Google by allowing Google to avoid taking certain costly actions."[150] I am unaware of any legal basis for using the cost savings associated with non-infringing alternatives for disgorgement, which I believe

---

[147] Forensic & Valuation Services Practice Aid – Calculating Intellectual Property Infringement Damages, AICPA, 2013, p. 22.

[148] http;//www.uniformlaws.org/shared/docs/trade%20secrets/utsa_final_85.pdf, Section 3, Damages.

[149] Expert Report of Dr. Leonard, February 8, 2016, p. 94.

[150] Expert Report of Dr. Leonard, February 8, 2016, p. 84.

is the appropriate measure of recovery in this matter. In fact, I understand this Court has previously ruled against Google on this very issue:

> *In his damages report, Dr. Cox opined: 'The ready availability of obviously acceptable non-infringing alternatives also provide [sic] basis that the 'element of profit' that is attributable to the allegedly infringed API claim contained in the Android framework is very small or zero.*
>
> …
>
> *As Dr. Cox makes clear in his report, the existence of 'multiple acceptable and effective' non-infringing alternatives 'at little or no additional cost' greatly reduces the lost licenses fees (Cox Report 61). This order finds this aspect acceptable. Not acceptable, however, is allowing the existence of non-infringing alternatives to reduce recovery of wrongful profits. This is a distinct remedy for the purpose of disgorgement. Non-infringing alternatives have nothing to do with this.*[151]

140. Akin to Dr. Cox, Dr. Leonard relies on a cost savings based analysis to reduce Oracle's recovery of Goggle's wrongful profits and determine the profit attributable to the Infringed Java Copyrights is very small.

141. What's more, each of the cost savings calculations Dr. Leonard relies on are fundamentally flawed and lack sufficient support. Dr. Leonard's cost saving scenarios are listed below, and my comments regarding each are provided in the sections that follow.

- Costs associated with switching to OpenJDK
- Costs associated with developer training in C/C++
- Costs associated with paying third party developers to develop Android apps in C/C++[152]

4.7.1 Avoidance of Costs Associated with Switching to Open JDK

142. Dr. Leonard states that the most appropriate measure of apportionment is the avoidance of costs associated with Google switching to OpenJDK, which he quantifies at $85,000.[153] The Murray, Jaffe, and Schmidt Reports each provide ample evidence for why OpenJDK was not a viable economic or technical alternative for Google and the record evidence shows that Google rejected it because it wasn't a commercially viable alternative. Below I summarized what I believe to be the most notable defects in Dr. Leonard's analyses in the section below:

- Dr. Leonard ignores the business and risk factors that Google and its OEMs and carriers would have considered.
- Dr. Leonard disregards that OpenJDK was not an adequate alternative due to performance and compatibility issues.

---

[151] Doc. 632 (Order Granting, in Part, Oracle's Motion to Exclude Portions of Leonard & Cox).

[152] Expert Report of Dr. Leonard, February 8, 2016, pp. 84-88.

[153] Expert Report of Dr. Leonard, February 8, 2016, p. 87.

- Dr. Leonard ignores that Google intentionally rejected the use of OpenJDK.

4.7.1.1 Dr. Leonard Ignores the Business and Risk Factors that Android and its OEMs Would Have Considered

143. Google faced a limited window of opportunity to launch Android, and if Google missed that window, Android faced being locked out of the mobile market. In addition, the payouts to Andy Rubin and his team connected to the Android acquisition were contingent upon reaching Milestones which included developing a working phone and securing a relationship with a major carrier; otherwise all future earn-out payments would be forfeited. For at least these reasons, Google did not want its platform subject to a "viral" license such as GPLv2-CE (e.g. OpenJDK).

144. Commercial exploitation under the GPLv2 is impractical and Google acknowledged the significant risk that GPL-related licensing presented to partner adoption and thus time-to-market imperatives. Under OpenJDK, there was substantial risk that OEM and mobile carriers would be required to open-source their modifications to the Android platform under the relevant GPL license.[154] For this reason, most, if not all, commercial entities implementing Java for use in a device have declined OpenJDK and instead done so under a commercial license.

145. In a November 12, 2006 email, Andy Rubin proclaimed "GPL license (sun's license doesn't work for us."[155]

146. In an August 11, 2007 email to Bob Lee, Brian Swetland and Dan Bornstein, all Google/Android engineers, Andy Rubin discusses the challenges posed to OEMs and mobile carrier by using GPL in Android and why Google would want to distance itself from GPL licenses as much as possible:[156]

> *... and as far as GPL-ing the VM, everything that is linked with the VM would get infected.*
>
> *The problem with GPL in embedded systems is that it's viral, and there is no way (for example) OEMs or Carriers to differentiate by adding proprietary works. We are building a platform where the entire purpose is to let people differentiate on top of it.*
>
> *Finally, Sun has a different license for its library for SE and ME. The SE library is LGPL, ME library is GPL. That means anything that links with the ME library gets infected. And the SE library is not optimized for embedded systems.*
>
> *Sun chose GPL for this exact reason so that companies would need to come back to them and take a direct license and pay royalties.*

---

[154] Expert Report of Gwyn Murray; http://arstechnica.com/uncategorized/2007/11/why-google-chose-the-apache-software-license-over-gplv2/.

[155] Trial Exhibit 154 (GOOGLE-01-0002454 – 457); Deposition of Anwar Ghuloum, December 9, 2015, p. 35 – 36, Trial Exhibit 230.

[156] Trial Exhibit 230; Exhibit 3 to the Deposition of Andy Rubin, April 5, 2011, GOOGLE-02-00020474-475 at 474.

Tricky, no? Why would we want to do anything to support this behavior? We want to distance ourselves as much as possible from Sun.

-andy

147. During the same time frame, Google publicly expressed concern over what using OpenJDK for Android code would mean to its OEMs. In a report on the 2008 Google I/O conference, one observer explained:

> *The first and main reason they give us for using Harmony instead of OpenJDK is the GNU license (GPL). Cell phone makers want to link proprietary value-add code directly into the system (into JVM-based apps. and/or service processes), and they do not want to worry about copyleft. Perhaps there is some education needed here about the classpath exception. (I know I don't understand it; maybe they don't either. And their license wonks appear to have a well-considered preference for Apache 2 over GPL+CPE.)*[157]

148. In 2008, Andy Rubin confirmed that Google deliberately chose to release Android under the Apache license in order to avoid the risks that a GPL license might impose on OEMs. In an interview to CNET in 2008, he explained:

> *The thing that worries me about GPL is this: suppose Samsung wants to build a phone that's different in features and functionalities than (one from) LG. If everything on the phone was GPL, any applications or user interface enhancements that Samsung did, they would have to contribute back. At the application layer, GPL doesn't work.*[158]

149. Similarly, the Open Handset Alliance website explains that the reason for choosing the Apache License for Android is to enable handset manufacturers to keep their innovations and differentiated features as closed source, which could not have been accomplished using other licenses.

> *Apache is a commercial–friendly [sic] open source license. The Apache license allows manufacturers and mobile operators to innovate using the platform without the requirement to contribute those innovations back to the open source community. Because these innovations and differentiated features can be kept proprietary, manufacturers and mobile operators are protected from the "viral infection" problem often associated with other licenses*[159]

150. Potential commercial licensees have always expected that they only had limited options: (1) they could pay for a commercial license for Java, including the APIs, which required the implementation to pass the Java Compatibility Kit (JCK) tests (also referred to as the "TCK" for Technology Compatibility Kit tests); (2) they could create an independent implementation of the specification, under a different license, which also required that the implementation passed the TCK; or (3) they

---

[157] https://blogs.oracle.com/jrose/entry/with_android_and_dalvik_at.
[158] http://www.cnet.com/news/why-oracle-not-sun-sued-google-over-java/.
[159] http://www.openhandsetalliance.com/android_faq.html.

could take the GPLv2-CE license, which relieved them of compatibility requirements, but mandated that they would have to release their source code under the terms and conditions imposed by that license.

151. Except for Google and its development of Android, most, if not all, commercial entities implementing Java for use in a device have done so under a commercial license. Google required the prompt buy-in of OEMs and carriers to be able to successfully launch Android. The threat of having to open-source proprietary modifications under OpenJDK's GPLv2-CE would have put those partnerships at risk, undermined Google's ability to achieve early entry, and threatened the ultimate success of Android.[160]

4.7.1.2 Dr. Leonard Neglects to Mention that Google Intentionally Rejected the Use of OpenJDK

152. Google intentionally *rejected* the use of GPLv2-based OpenJDK.[161] If implementing OpenJDK for $85,000 was a viable alternative, economic logic dictates that Google would have done so.[162] Google faced known risks and made "enemies" using the Infringed Java Copyrights without a license. Therefore, it does not make sense for Google to have borne those risks and make those enemies, if they could easily have been avoided for $85,000. Assuming Google seeks to maximize long-term profits, this suggests that OpenJDK was not, and has not been, an economically favorable choice for Google.

153. After bringing Android to market, Google acknowledged the looming lawsuit it would surely face because of its unauthorized copying of Java in Android. In August 2010, Mr. Lindholm wrote to Andy Rubin that Google's founders Larry Page and Sergey Brin had asked him "to investigate what technical alternatives exist to Java. . . " Lindholm added that his team had "been over a bunch of [alternatives to Java]" and that "they all suck." His conclusion was "that [Google] need[ed] to negotiate a license for Java . . . "[163]

4.7.2 Avoidance of Costs Associated with Developer Testing

154. Dr. Leonard's calculation of the $2.3 million in avoided costs associated with training developers in C/C++ is incorrect for at least the following reasons, which I describe further in the sections that follow:

- Dr. Leonard incorrectly conflates replacing the Java Community with replacing the Java language
- Dr. Leonard exaggerates the utilization of NDK and provides no support for C/C++ as a reasonable alternative

---

[160] GOOGLE-02-00020474; Expert Report of Dr. Kemerer, February 8, 2016, ¶ 258.
[161] Trial Exhibit 154.
[162] http://venturebeat.com/2015/12/29/google-confirms-next-android-version-wont-use-oracles-proprietary-java-apis/.
[163] GOOGLE-12-10000022.

- Dr. Leonard greatly underestimates the full cost to replace Java-based apps in the Google Play store

4.7.2.1 Dr. Leonard Incorrectly Conflates Replacing the Java Community With Replacing the Java Language

155. By limiting his analysis to the costs associated with replacing the Java language contained in certain mobile apps, Dr. Leonard inherently mischaracterizes the use made of the Infringed Java Copyrights by Google. Google derived great value not only through the use of the Java platform, but more importantly through the poaching of the vast Java Community, that included OEMs, carriers and developers. [redacted].[164] [redacted].[165] [redacted].[166] In an e-mail from Andy Rubin to Larry Page on October 11, 2005, Mr. Rubin acknowledged that Java has advantages and that it was "the #1 choice for mobile development."[167]

156. A 2006 Sun presentation illustrates the widespread success of the Java Community that Google obtained through its use of the Infringed Java Copyrights, highlighting the platform's ubiquity and potential for future growth among numerous markets.

---

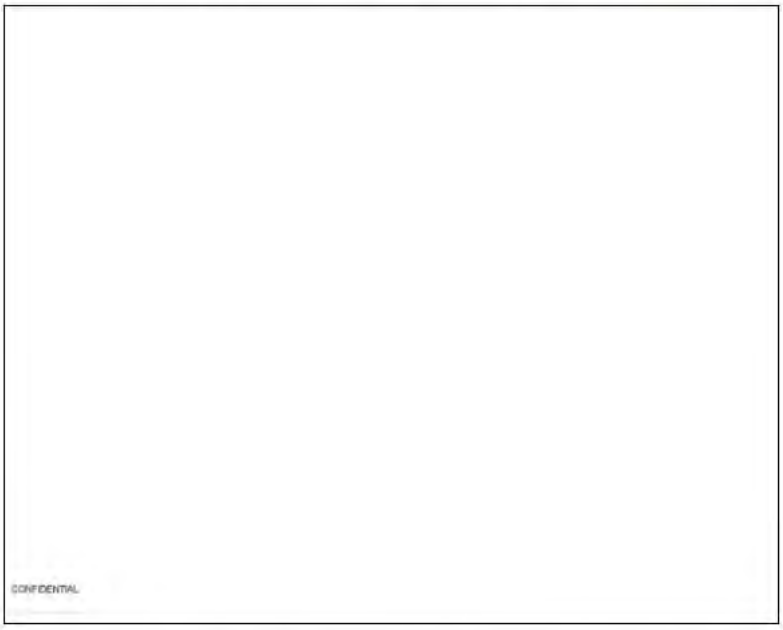
[164] OAGOOGLE3000000021-024.

[165] OAGOOGLE3000000021-024; http://www.prnewswire.com/news-releases/sun-strengthens-lead-in-worldwide-mobile-data-services-with-java-72506432.html.

[166] OAGOOGLE0004260166-187 at 167.

[167] GOOGLE-01-00019527.

Figure 6
Depiction of Java Community from Sun Presentation[168]



157. Google faced a closing window of mobile opportunity to avoid being locked out. The Infringed Java Copyrights provided Google a powerful means of accessing dozens of OEMs and mobile carriers and millions of Java developers that Google desperately needed. Google's internal emails confirmed that the alternatives "all suck"[169] and Android needed access to Java and its developer community[170] and that the Infringed Java Copyrights would attract them to the Android platform.[171]

4.7.2.2 Dr. Leonard Exaggerates the Utilization of NDK and Provides no Support for C/C++ as a Reasonable Alternative

158. Dr. Leonard mischaracterizes Android NDK as a popular platform that accounts for at least half of Android app development. Android NDK is a toolset that allows a developer "to implement parts of your app using native-code languages such as C and Cs++."[172] I understand that the NDK is not even encouraged as the Android developers guide states "[T]he NDK is not appropriate for most novice Android programmers, and has little value for many types of Android

---

[168] GOOGLE-01-00018140 at 143. Incidentally, the cover email dated January 31, 2006 of this document shows that Andy Rubin was in possession of this Sun presentation and indeed asked for the information continued therein.
[169] GOOGLE-12-10000022.
[170] Deposition of Daniel Bornstein, May 16, 2011, p. 47.
[171] Deposition of Daniel Bornstein, May 16, 2011, pp. 47-49.
[172] http://developer.android.com/tools/sdk/ndk/index.html.

apps. It is often not worth the additional complexity it inevitably brings to the development process."[173] In fact, Google actually discourages developers from using NDK outside specific use cases such as game engines, signal processing and physics simulation. Google also states that "the NDK will not benefit most apps" and that using "native code on Android generally does not result in a noticeable performance improvement, but it always increases your app complexity. In general, you should only use the NDK if it is essential to your app – never because you prefer to program in C/C++."[174]

4.7.2.3 Dr. Leonard Greatly Underestimates the Full Cost to Replace Java-Based Apps in the Google Play Store

159. Dr. Leonard's assumption that a $715 course in a foreign programming language would be sufficient for a Java programmer to build and maintain Android's most downloaded apps is nonsensical. An oversimplified yet not unreasonable analogy would be a unilingual student taking a single foreign language class, and then assuming the student would be able to create the best-selling works of that language. To develop large scale, complex, and highly used apps typically takes years.[175] Thus, minimizing the effort of learning C/C++ to a single course and cost of $715, as Dr. Leonard has done, is not reasonable.

160. Dr. Leonard's assumption that 3,000 developers would replace the collective contribution to the Android Platform of millions of Java developers is also not reasonable. In order to be competitive in the mobile market with companies such as Microsoft, Andy Rubin acknowledged the importance of leveraging "millions" of Java developers in order to meet the critical window of mobile opportunity.[176] To suggest that Android would achieve a similar outcome with a developer community of a few thousand is illogical.

161. In addition to relying on a flawed methodology and assumptions, Dr. Leonard's actual calculation also appears to be flawed. He estimates that approximately 3,779 developers are responsible for the top 100 downloaded apps on Android for the years 2008 – 2015. After making adjustments for apps written using NDK and those on multiple platforms, he multiplies by 1.6 to account for the number of programmers per app developer. 1.6 programmers per app is likely a drastic understatement. To put this number into perspective, King Digital (recently acquired by Activision Blizzard), a developer of just four of the top 100 apps on Dr. Leonard's list has 1,600 employees.[177] Many of those employees are developers, which was a key reason Activision acquired King, as they desired mobile expansion.[178] He derives the programmers per app ratio from the Google

---

[173] http://developer.android.com/ndk/guides/index.html.
[174] http://developer.android.com/tools/sdk/ndk/index.html.
[175] http://www.coderanch.com/t/507541/java/java/long-good-Java.
[176] Expert Report of Dr. Jaffe, February 8, 2016, ¶ 162.
[177] https://www.macroaxis.com/invest/ratio/KING--Number-of-Employees.
[178] https://fortune.com/2015/11/03/activision-blizzard-king-digital/.

Developer Challenge in 2008, in which only one of the applications from the Developer Challenge are among the highly rated applications that he listed in Exhibit 2i.[179]

162. Finally, Dr. Leonard's analysis ignores that Android would need hundreds of thousands of apps available to be attractive to developers and consumers. By limiting his analysis to Java-enabled apps in the top 100, he fails to value the hundreds of thousands of other Java-enabled apps available on Google Play. Dr. Leonard is misguided to suggest that replacing the top few hundred apps would still enable Android to be competitive with platforms such as iOS, which today offer over a million apps.

4.7.3 Avoidance of Costs Associated with Paying Third Party Developers to Develop Android Apps

163. Dr. Leonard estimates the avoided costs associated with paying third party developers to develop the "most used" Android apps in C/C++ is $23 million to $100 million. His use of the cost approach and C/C++ as a reasonable alternative is improper for the reasons mentioned above as well as those listed below, which I describe further in the paragraphs that follow:

- Dr. Leonard's analysis is based on a marketing strategy that Google determined didn't work and wasn't considered to be an effective mechanism for app development
- Dr. Leonard underestimates the full costs of developing a mobile app
- Dr. Leonard only considers the cost of development for 1,000 apps and ignores the remaining 1.6 million apps currently offered by Google Play

4.7.3.1 Dr. Leonard's Analysis is Based on a Marketing Strategy that Google Determined Didn't Work and Wasn't Considered to be an Effective Mechanism for App Development

164. Dr. Leonard's analysis partly relies upon testimony from Reto Meier, Developer Relations employee at Google. Mr. Meier testified that Google considered financially incentivizing developers to build Android apps, but did not do it because it was not effective, particularly for big brands (e.g. developers of highly used apps). Specifically, Mr. Meier testified that Google elected not to undertake the plan because "it wasn't considered to be an effective mechanism for incentivizing app development."[180]

165. An October 2010 email between Mr. Meier and several Developer Relations employees at Google explained that incentivizing third party app development was "the regular MO for both MS and Nokia" and that "Microsoft and Nokia are not only funding (and managing) the app development,

---

[179] Expert Report of Dr. Leonard, Exhibit 2i, "Free and Paid Apps Appearing on Daily Top 100 Download Lists, June 2015".

[180] Deposition of Reto Meier, December 11, 2015, p. 71.

they're also paying companies above and beyond development costs for the privilege" adding, however, that "such a strategy is not self-sustaining in the long term."[181] In the same email chain, Mr. Meier compared incentivizing third party developers to "throw[ing] some money out the window and see if it comes to something."[182]

4.7.3.2 Dr. Leonard Underestimates the Full Costs of Developing a Mobile App

166. In his testimony, Mr. Meier stated that such a program wasn't compelling to developers and ineffective because it "may have offsetted [sic] the initial upfront development costs, but the long-term, ongoing development maintenance and – support of a product would continue to cost additional resources for the company….there's significant risk that they wouldn't continue to develop, evolve, and ensure that it continued to be a high-quality app would have, in the longer term been a negative for the platform, and, additionally, the offset, in terms of ROI to developers, just wasn't that compelling…If we are talking about big brands, these aren't huge amounts of money, and their interest, from my recollection, was more of a longer-term ROI rather than an initial requirement to be able to offset the initial development." [183]

167. Dr. Leonard ignores the long term costs required to develop and maintain a mobile app and instead basis his calculation only on estimated up front development costs of $25,000 - $100,000. Industry experts have indicated that up front development costs are only the "the tip of the iceberg".[184] A November 2014 Kinvey Report based on a survey of CIOs and Mobile Leaders found that mobile app development can be "costly, slow and frustrating" and that for development costs alone "…18 percent say they spend from $500,000 to over $1,000,000 per app, with an average of $270,000 per app."[185] Furthermore, the Google presentation that Dr. Leonard relies upon specifies that up front development costs for games is $500,000.[186] In 2014, mobile games accounted for approximately 90% of Android's app revenue, however, Dr. Leonard caps his cost-per-app range at $100,000.[187]

168. Dr. Leonard also incorrectly conflates the up front development cost for average mobile apps with that of the "most used apps". Highly used apps are likely to cost much more to develop and maintain than an average mobile app. The development of apps with millions of users requires the apps to be stable, robust, and tested for quality and failure under numerous test cases. I understand that in any coding process, time for debugging, testing, and ensuring quality usually takes much longer than simple code writing. There are numerous code processes and software quality initiatives that are required to ensure code works under a number of circumstances and test

---

[181] Exhibit 5025 to the Deposition of Reto Meier, December 11, 2015, GOOGLE-37-00023782-785 at 783-784.
[182] GOOGLE-37-00023782-785 at 782.
[183] Deposition of Reto Meier, December 11, 2015, p. 72.
[184] http://www.formotus.com/14018/blog-mobility/figuring-the-costs-of-custom-mobile-business-app-development.
[185] http://www.formotus.com/14018/blog-mobility/figuring-the-costs-of-custom-mobile-business-app-development.
[186] Exhibit 5024 to the Deposition of Reto Meier, December 11, 2015, GOOGLE-03-00007402 at 462.
[187] http://www.androidauthority.com/app-annie-2015-app-retrospective-668731/

conditions.[188] It would be unreasonable to suggest that the full costs associated with building and maintaining a highly used app are limited to $25,000 - $100,000.

4.7.3.3 Dr. Leonard Only Considers the Cost of Development for 1,000 Apps and Ignores the Remaining 1.6 Million Apps Currently Offered by Google Play

169. As with the previous analysis, Dr. Leonard assumes the other large portion of the 1.6 million apps that are Java-based and currently available on Google Play are irrelevant to Google's causally connected profit. By only quantifying the cost to develop 1,000 apps, Dr. Leonard neglects to value the cost to develop the additional hundreds of thousands of Java-based apps available for Android. To assume Android would be equally successful with only a thousand available apps completely ignores the impact of opportunity costs.

170. By applying Dr. Leonard's calculation to the total number of 1.6 million available apps in Google play, rather than the most used apps, approximately 560,000[189] Java-based apps would need to be developed in C/C++.[190] According to Dr. Leonard's average cost of development, that would cost Google $14 billion to $56 billion.

5. RESPONSE TO DR. LEONARD'S LOST PROFIT OPINIONS

171. As stated previously, in spite of the rebuttal opinions put forth in the Leonard Report, the opinions expressed in my Initial Report regarding Oracle's lost profits remain unchanged. In response to the opinions put forth by Dr. Leonard and as further support for the opinions reflected in my Initial Report, I note the following:

---

[188] https://www.atlassian.com/landing/software-testing/.

[189] Determined by using Dr. Leonard's estimate of 35% of Java based apps times 1.6 million available apps on Google Play.

[190] http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/; Expert Report of Dr. Leonard, February 8, 2016, footnote 277. 1.6 million apps in Google Play store x 35% of Google Play apps that are non-Google apps, non-C++ apps, or apps not by multi-homing developers = 560,000.

- Dr. Leonard's zero lost profits opinion is unreasonable and inconsistent with the evidence
- Dr. Leonard's alternative lost profit opinions are speculative and unreliable
- Dr. Leonard inappropriately compares lost profits to Sun/Google negotiations
- Dr. Leonard improperly considers Java to be "stagnant"
- Dr. Leonard overstates the impact of the recession on the mobile industry
- Dr. Leonard fails to account for Sun's dominance among carriers and OEMs (for example) in the feature phone market
- Dr. Leonard fails to consider Sun's ability to transition its dominance into the smartphone market
- Dr. Leonard does not address Sun's relationship with Nokia

5.1 Dr. Leonard's Zero Lost Profits Opinion is Unreasonable and Inconsistent with the Evidence

172. Dr. Leonard opines that Oracle's lost profits are "zero" and that Google's infringement of the Java Copyrights did not result in the decline in Java ME revenue.[191] However, Dr. Leonard's opinions fail to consider important facts and circumstances that render an opinion of zero lost profits unreasonable. The information presented in my Initial Report, and further detailed in this responsive report, provide strong support for my opinion that Google's infringement of the Java Copyrights caused Sun, now Oracle, to lose profits. Not only did Google's illegal use of the Infringed Java Copyrights negatively impact Sun and Oracle's ability to enforce and renew Java ME licenses, it also adversely impacted Sun's ability to capitalize on the emerging smartphone market.

173. To arrive at a zero lost profits opinion, Dr. Leonard ignores evidence that directly relates to lost Java ME license revenue as a result of Android.[192] [redacted][193] [redacted][194] Similarly, Henrik Stahl testified:

[redacted][195]

[191] Expert Report of Dr. Leonard, February 8, 2016, p. 11.
[192] OAGOOGLE0000799926.
[193] OAGOOGLE0000799926.
[194] OAGOOGLE0000457616-617 at 617.
[195] Deposition of Henrik Stahl, January 14, 2016, pp. 162-164.

[redacted][196]

174. In an attempt to discredit my Java ME lost profits analysis and thus provide support for his zero lost profits opinion, Dr. Leonard improperly suggests that my lost profits opinion is overstated because the losses per Android device profit is higher in earlier years than it is in later years.[197] This opinion is fundamentally flawed as it suggests Sun's losses in the early years would immediately tie to phone sales. Once Android was chosen by OEM's/carriers there would be a delay as to when the Android phones would enter the market. Dr. Leonard's per-unit calculation also inappropriately suggests that the relationship between Android units and lost Java ME profits should be linear throughout the entire damages period. However, Android does not have a static relationship with Java ME whereby a unit of Android on the market causes a specific level of Java ME lost profits. Sun lost entire business relationships, it lost the opportunity to compete in the smartphone space, and it lost the ability to significantly monetize Java through its historic licensing model.

175. Dr. Leonard also incorrectly argues that a decline in the amount of Java ME revenue per feature phone suggests that Android did not cause Java ME lost profits.[198] He reasons that, since Oracle was unable to earn consistent Java ME revenue per feature phone, a decline in Java ME could not be the result of Google's infringement because Android does not compete with feature phones.[199] First, Android does in fact compete with feature phones.[200] Even if Android were used predominantly on smartphones, it is incorrect to assume that it does not compete with other mobile devices, as Dr. Leonard suggests. Rather Android entered the market and gained rapid success precisely because it was able to successfully and effectively compete with existing technology on the market, including feature phones. Furthermore, feature phones and smartphones are not entirely distinct from one another as the functionality exists on a continuum, and feature phones continue to dominate the international mobile device market.[201] Dr. Leonard's opinions in this regard are further flawed, given my understanding that Android has been used on feature phones.[202] Dr. Leonard also ignores Sun's attempts to transition from feature phones to smartphones as evidenced by its acquisition of SavaJe in 2007.[203] [redacted]

---

[196] Deposition of Henrik Stahl, January 14, 2016, pp. 162-164.

[197] Expert Report of Dr. Leonard, February 8, 2016, p. 136. Exhibits 4a and 4b to the Expert Report of Dr. Leonard, February 8, 2016.

[198] Expert Report of Dr. Leonard, February 8, 2016, p. 136-137.

[199] Expert Report of Dr. Leonard, February 8, 2016, p. 136-137.

[200] http://www.cnet.com/news/android-and-the-future-of-feature-phones/.

[201] http://www.philstar.com:8080/telecoms/2013/04/06/927330/asha-blurring-lines-between-feature-phones-smartphones; http://qz.com/418769/theres-still-plenty-of-money-in-dumb-phones/.

[202] http://www.cnet.com/news/android-and-the-future-of-feature-phones/.

[203] OAGOOGLE0006231006-033 at 025; OAGOOGLE0000473609-612; OAGOOGLE0000424812-813 at 812.